NAG 2-513
IN-73-CR
91204
P-17

Analyzing High Energy Physics Data Using Database Computing: Preliminary Report*

Andrew Baden, Chris Day,
Robert Grossman, Dave Lifka, Ewing Lusk,
Edward May, and Larry Price

December, 1991

Abstract

We describe a proof of concept system for analyzing high energy physics (HEP) data using database computing. The system is designed to scale up to the size required for high energy physics experiments at the Superconducting SuperCollider (SSC) laboratory. These experiments will require collecting and analyzing approximately 10 to 100 million "events" per year during proton colliding beam collisions. Each "event" consists of a set of vectors with a total length of approximately one megabyte. This represents an increase of approximately 2-3 orders of magnitude in the amount of data accumulated by present large HEP experiments. The system is called the HEPDPC System (High Energy Physics Database Computing System). At present, the Mark 0 HEPDBC System is completed, and can produce analysis of HEP experimental data approximately an order of magnitude faster than current production software on data sets of approximately 1 GB. The Mark 1 HEPDBC System is currently undergoing testing and is designed to analyze data sets 10 to 100 times larger.

1 Introduction

In this paper, we describe a proof of concept system for analyzing high energy physics (HEP) data using database computing. The system is designed to scale up to the size required for high energy physics experiments at the Superconducting SuperCollider (SSC) laboratory. These experiments will require collecting and analyzing approximately 10 to 100 million "events" per year during proton colliding beam collisions. Each "event" consists of a set of vectors with a total

*For more information, contact Robert Grossman, Laboratory for Advanced Computing, m/c 249, University of Illinois at Chicago, Box 4348, Chicago, IL 60680, grossman@math.uic.edu.

length of approximately one megabyte. This represents an increase of approximately 2-3 orders of magnitude in the amount of data accumulated by present large HEP experiments.

The system is called the HEPDPC System (High Energy Physics Database Computing System). At present the Mark 0 HEPDBC System is completed, and an early version of the Mark 1 HEPDBC System is undergoing testing. The Mark 0 version of the system was built using commercial software and is designed to analyze efficiently approximately 1-10 GBs of data. The Mark 1 version of the system is being written in C++, with hooks into commercial databases, and is designed to analyze 10-100GBs of data. Later versions of the system are in the planning stages and will scale up to the TB range and beyond.

2 Background

2.1 Scientific databases

A database provides access to data via queries regarding the data itself, and not its physical or logical location. The relational model is a simple and powerful model suited to many business and commercial applications, especially those applications making use of simple data in a fixed format. Applications, such as computer aided design, computer aided manufacturing, office information systems, and artificial intelligence, have resulted in extentions to the relational model [5], and new models, such as the object-oriented data model [10]. More recently, applications requiring the management of scientific and engineering data, such as time series, satellite data, DNA sequence data, seismic data, and collider physics data have resulted in attempts to define a data model suitable for these types of applications [8], [12], [14] and [15]. Just as there is currently no agreed upon definition of an object-oriented data model, there is also no agreed upon definition of a data model for scientific and engineering computations. In § 4, we briefly describe the provisional data model we are using.

Issues which turned out to be important in our system are somewhat different than issues of importance in traditional business applictions or more recently applications in CAD/CAM and articifical intelligence. For our application in HEP, the main requirements of our system are:

- the ability to query very large amounts of data (expected to greater than 1PB), stored in a hierarchical storage system [13]
- the ability to efficiently execute numerically intensive queries specified by Fortran or C functions
- the ability to support large, complex objects
- a mechanism for working with *derived data*, that is data that is derived from other data via some computation, and the ability to dynamically

determine which data should be precomputed or computed in the background.

Issues of importance in other scientific databases [8], such as the level of interpretation of the data, the intended analysis of the data, the source of the data, and the use of metadata do not present major problems in this application.

2.2 Traditional data analysis in HEP

In this section, we review how HEP data is currently analyzed following [3]. At present, the CDF colliding beam experiment at the Fermi National Accelerator Laboratory (FNAL) measure the radiation products from the collision of particle beams at rates of up to 285 kHz. Collisions, or events, are recorded in detectors which provide $\sim$ 150kbytes of digital information. However, only a small fraction ($\sim$ 1Hz) of the total number of events are recorded on magnetic tape for data analysis. The computing environment can be broken down into the following categories:

- **Online**: In the *online* stage, events which warrant scientific investigation are identified through a sequence of several decisions (a mixture of hardware and software processing), each requiring greater execution time. A decision to keep a particular event, called a *trigger*, results in the corresponding raw data being collected and written to magnetic tape.
- **Production**: In the *production* stage, the raw data is processed by computer codes (production codes) which reconstruct the digital data words in order to identify the number and type of the particles which characterize each event. Specifically, the result of the production codes are dimensional quantities such as energy, mass, momentum, orbits, event topology, and *etc*. These data summaries are written to some medium, usually magnetic tape, for further analysis, referred to as *data summary tapes* (DSTs).
- **Analysis**: In the *analysis stage*, computations are performed on the data in the data summary tape. For instance, such computations can be used to discover (or verify) correlations between certain types of events, or measure an average of a quantity over a particular set of events.

Most HEP computing schemes store data in sequential records using memory management software systems written at HEP laboratories. Data are analyzed via computer programs which are written mostly in Fortran, access events from storage as records, read it into common memory, and provide hooks for users to apply custom-written subprograms. Each of these programs are compiled and linked to a standard set of libraries. Linking, at present, is not usually done interactively.

Once the analysis program is built, it is made to loop over a number of events. For each event the entire record is read from storage, and those parts which are relevant to the particular analysis (and very few analyses use all of the data in a single event) are used by the user subprogram. For instance, a typical session would consist of:

1. Read in each event. If the events are on tape, access the tape first. For each event:
   - require certain global event characteristics;
   - calculate quantities from each event (*e.g.* the presence of a certain number of a certain type of particles);
   - require the particles to have certain characteristics;
   - fill histograms or scatter plots of distributions.
2. If desired, save events passing requirements onto disk or tape.
3. Report results of calculations, histograms, plots, regressions, *etc.* at the end of session.

Although this procedure is straightforward, inefficiencies arise since the procedure is repeated several times to produce sequentially smaller datasets for analysis by different working groups. Also, although it is often convenient to produce additional datasets, this is not usually done, due to the expense and difficulty of doing so. The critical point is that access to the data is via entire event records, and in a sequential manner, *i.e.* events are read as atomic.

Expectations at the SSC are that there will be an increase relative to the largest ongoing HEP experiments to date of about ~ 1 order of magnitude in the size of each event (to 1 Mbyte/event), $\sim 2-3$ orders of magnitude in the number of events collected for analysis (from 1 to 10-100 Hz to tape), and ~ 1 order of magnitude in the number of physicists who will be accessing the data (~ 1000 physicists).

2.3 Prior Work

Recently, the data analysis stage has been made easier by using an interactive data analysis program called PAW and a data access package called Adamo, both of which were developed at the European Organization for Nuclear Research (CERN) in Geneva, Switzerland. The program PAW (Physics Analysis on Workstations) resembles a spreadsheet program and incorporates an inline Fortran interpreter. The Fortran interpreter eliminates the need to need to link the large Fortran programs previously used to analyze the data. The interpreter has been found to be fast enough, since most of the time spent when running in the interpreter mode is for I/O. Data analysis with PAW must be preceded with a transformation of the data to a limited relational tuple. However, once the data is transformed, the program allows one to interactively identify events with computed quantities passing certain thresholds (*cuts*) and to study correlations and anti-correlations of the particles and events. The program also supports many standard statistical operations, such as constructing histograms, scatter plots, curve of best fit, and *etc.* However, the system has well-defined data structures which do not allow dynamic changes, and require a preprossing to transform existing HEP data (which is by nature dynamic).

Adamo is a data structuring package, developed for the Aleph experiment at CERN, and written in Fortran [1]. It is based on relational tables and allows the modeling of the complex event structures used in HEP. However, its modeling is limited to a single event at a time, that is, a complete event is read into memory and restructured into Adamo tables. The program can then access portions of the event through relational queries. While this model is good for production code, where all information about a single event is needed at once and information about other events is irrelevent, it is not appropriate for analysis, which involves the collection of statistics from a large number of events, each statistic based on only a subset of each event's information.

3 The Mark 0 HEPDBC System

Using a proof-of-concept system (*the Mark 0 HEPDBC System*), we analyzed approximately 0.6 GB of data from the CDF experiment at Fermi National Laboratory to produce the mass histogram of psi-candidates in Figure 1, which reveals a psi particle at approximately 3 GeV (the square of the mass is plotted in the histogram). The queries producing this hisogram required approximately 30 seconds of CPU time and 41 seconds of elapsed time on a Sun Microsystems SparcStation 1. For comparison, the program ACDUMPEV, which is currently in production use at Fermi National Laboratory, required approximately 14 minutes and 12 seconds of CPU time and 1 hour and 12 minutes of elapsed time on a DEC VAX. These times are given for rough illustrative purposes only: there a variety of reasons while the comparasion cannot be taken at face value. We expect, though, that a naive implementation of our system would result in approximately an order of magnitude speedup in the analysis of HEP data, and that more sophisticated implementations would result in substantially greater speedups. Moreover, our system is designed to handle the larger amounts of data expected by the next-generation collider beam experiments.

3.1 System Architecture

We planned an architecture which would accommodate the development of the system over a period of several years. This architecture is shown in Figure 2. An important objective of this design is its modularity: research and prototype development of various parts of the system can proceed relatively independently once certain interfaces are defined. The right-hand edge of the diagram represents the current analysis path for HEP data. We have defined an external format (item 3) and begun the translation process (item 2) to make a relatively small but interesting set of data available in an easily manipulated format. This has made it possible to conduct preliminary experiments with commercially available database systems in order to compare the ease and speed of doing a typical analysis in multiple ways. Some preliminary results are given below. Research is continuing on the exact form that the system-independent queries and output should take. We have constructed a simple front-end (Figure 3) that allows interactive access to the database systems and interactive manipulation of histogram data. In Figure 3, we see one histogram identifying the psi particle and another identifying the Z particle. That data is from the query described in Section 3.2.

The data we analyzed was first translated from a compressed VAX binary format into a neutral format consisting of ASCII data arranged in C-structures. This data was then loaded into a prototype of the HEPDBC system written on top of Sybase and into a prototype of the HEPDBC system written on top of Object Design's Object Store. A front end then allowed the user to query the system and to produce the histogram plots. The right hand edge of the figure illustrates the conventional means of analyzing HEP data using the Analysis Control Program and PAW described in Section 2.2. This figure also illustrates parts of the system under development, but not yet complete, including a suitable data model [4] and a suitable query language for analyzing scientific data. Also under testing is a stand alone version of the HEPDBC System (*the Mark 1 HEPDBC System*) that is independent of any commercial system and written in C++.

3.2 An Example

In this section we give an example of a representative physics analysis query and compare its implementation in the three prototype environments. Stated in English, the query is as follows:

1. Find all records which are events and have exactly 2 muon tracks.
2. Find the calibrated energy and momenta and form the 4-vectors for each muon track.
3. Calculate the effective 2 particle mass from the summed 4 vectors.
4. Order the result for histogramming.

Currently, this type of query is carried out by running a Fortran program of more than 300 lines. Both the relational and object-oriented database versions of this query are simpler and faster. We give here the C++ code necessary to express the above query against the data stored in an Object Store database, and also the SQL query required to express the above query against a Sybase database.

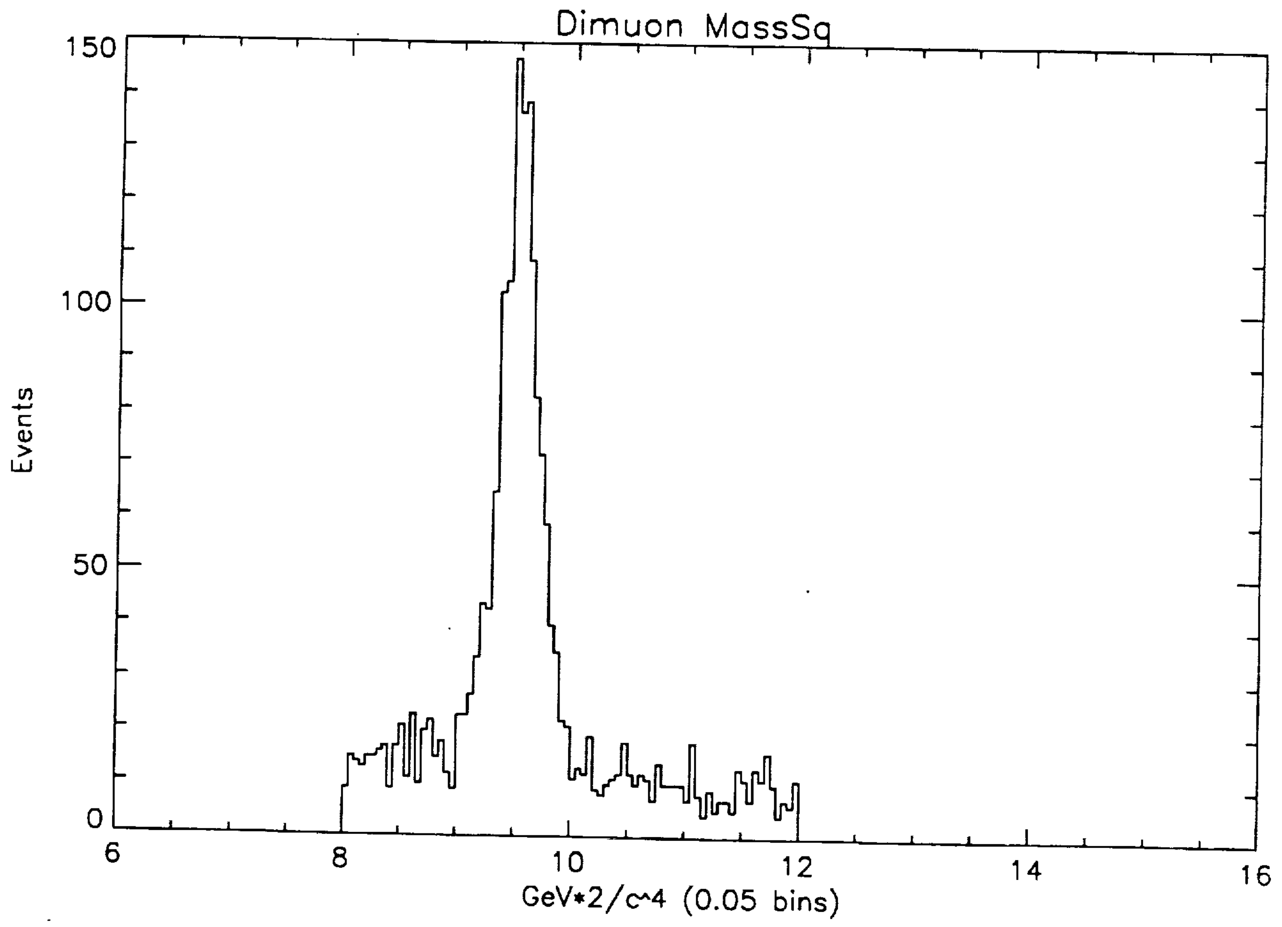

Figure 1: Histogram revealing the presence of a particle with energy ≈ 3 GeV.

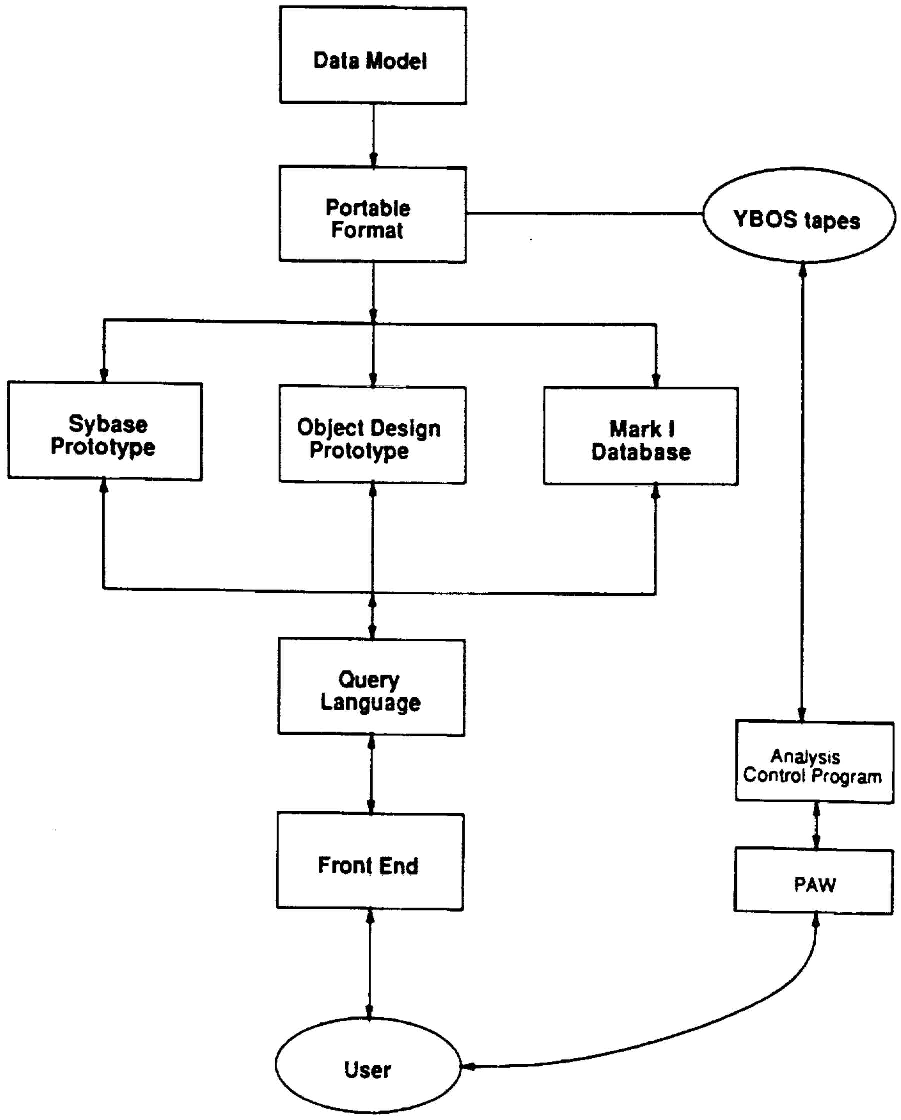

Figure 2: Prototype System Architecture

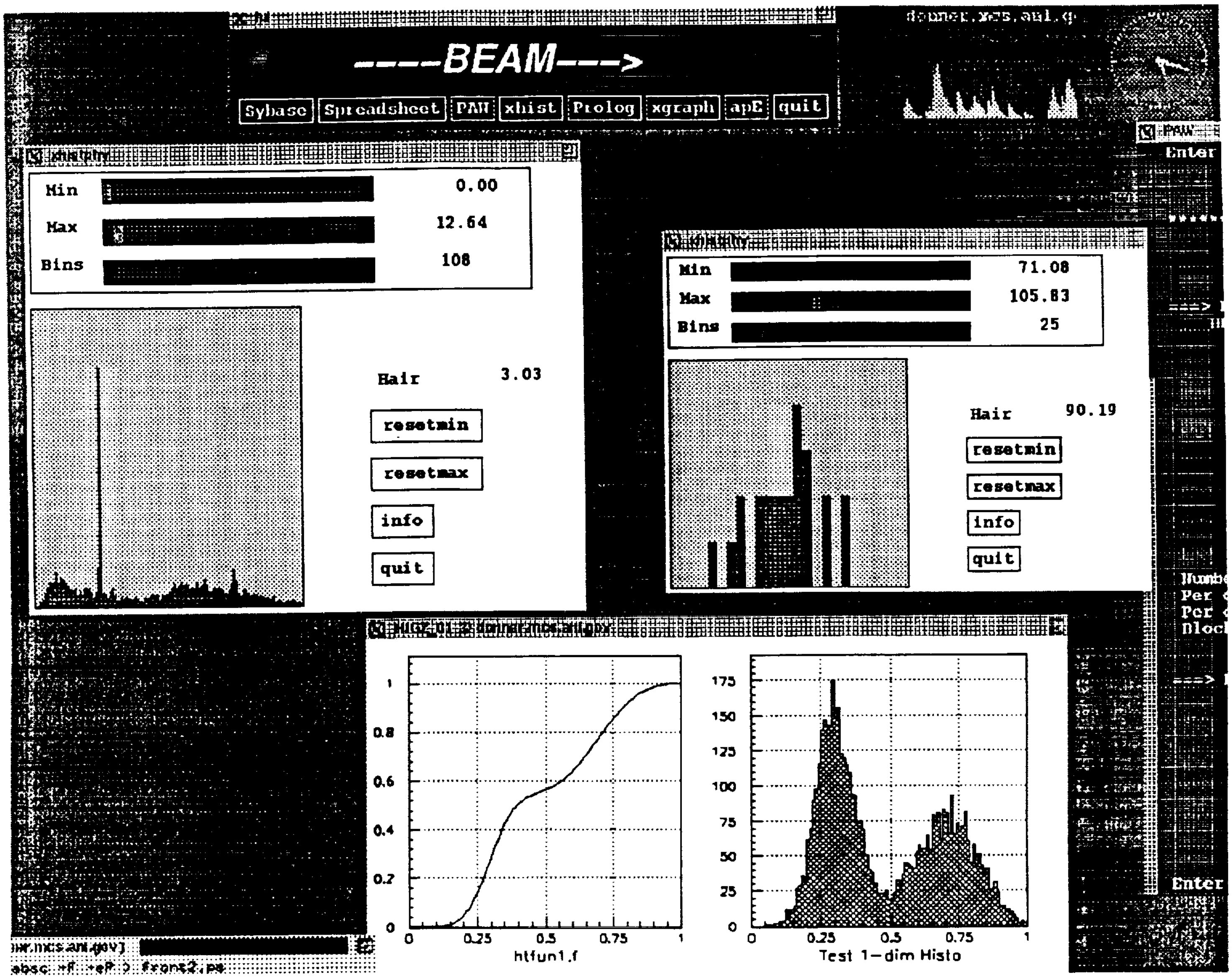

Figure 3: Front end to HEPDBC System.

```
extern database *dbTEvent;
extern database *dbTLepton;

class TEvent;
class TLepton;

class TEvent {
public:
  persistent <dbTEvent> os_Set<TEvent*> extent;
  int runNumber;
  int eventNumber;
  float vertex;
  TLepton *lepton1;
  TLepton *lepton2;
  TEvent() {extent.insert(this);}
  ~TEvent();
  friend ostream& operator << (ostream&, TEvent&);
  friend istream& operator >> (istream&, TEvent&);
};

class TLepton {
public:
  persistent <dbTLepton> os_Set<TLepton*> extent;
  float p[4]; // 4-momentum
  float charge;
  TLepton() {extent.insert(this);}
  ~TLepton() {extent.remove(this);}
  friend ostream& operator << (ostream&, TLepton&);
  friend istream& operator >> (istream&, TLepton&);
};
```

Figure 4: Definitions of classes TEvent and TLepton.

```
os_Set<TEvent*> neutralEvents;
os_Set<TDiLepton*> neutralDiLeptons;
os_Set<TDiLepton*> psiCandidates;
os_index_path massSqPath;

ofstream result("massSq.dat");

dbTEvent = database::open("/chris/Psi/dbPsi-simple", 1);
dbTLepton = database::open("/chris/Psi/dbPsi-simple", 1);
dbTransient = database::get_transient_database();

do_transaction() {
    cout << ' Number of events = "
 << TEvent::extent.cardinality() << "\n";
    neutralEvents =
TEvent::extent[: this->lepton1->charge == -this->lepton2->charge :];
    cout << " Number of neutral events = "
 << neutralEvents.cardinality() << "\n";
    foreach(TEvent* neutralEvent, neutralEvents) {
neutralDiLeptons |=
    new(dbTransient) TDiLepton(*neutralEvent->lepton1,
       *neutralEvent->lepton2);
    }
}
    massSqPath = pathof(TDiLepton*, massSq);
    neutralDiLeptons.add_index(massSqPath, os_Collection::ordered);
    psiCandidates =
neutralDiLeptons.query("TDiLepton*",
       "(7.0 <= massSq) && (massSq <= 15.0)",
       dbTransient);
    cout << " Number of Psi candidates = "
 << psiCandidates.cardinality() << "\n";
    foreach(TDiLepton* psiCandidate, psiCandidates) {
result << psiCandidate->massSq << "\n";
    }
```

Figure 5: A query computing Psi candidates.

```
Number of events = 7639
Number of neutral events = 5138
Number of Psi candidates = 1098
```

Figure 6: Result of Psi query.

```
select count(*) from dimuons2
select run_number, event_number, e=lepton_1_energy+lepton_2_energy,
   pz=p1z+p2z, py=p1y+p2y, px=p1x+p2x
into two_particle_4v from dimuons2 where charge_1+charge_2=0
select count(*) from two_particle_4v
select run_number, event_number, m=sqrt(e*e-pz*pz-py*py-px*px)
into two_particle_mass from two_particle_4v
select run_number, m from two_particle_mass
   where m > 70 order by m asc
```

Figure 7: SQL version of the Psi query.

4 Scientific Data Model

The Mark 0 HEPDBC System was designed using an entity-relationship model [7] and using a scientific data model developed for this purpose (more fully described in [4], and briefly reviewed here. The use of an entity-relationship model proved helpful in developing an understanding of the data and organizing for loading into the various prototype databases. One satisfying result of the analysis was that many of the subsets of data (called "YBOS banks" in the HEP community) emerged as entities or parts of entities in the ER model, which was developed through discussions between physicists and computer scientists with no regard for current data structures. Given that the YBOS system was designed intelligently, this served as a valuable check on the ER-model. A preliminary ER-diagram is given in Figure 8.

The scientific data model we are developing supports objects, attributes, methods, collections, inheritance, and a notion of data-dependency. Its salient features are:

objects The basic entities are objects. Objects are instances of classes. Examples of classes include Integers, Strings, Vectors, FourVectors, Events, and ParticleCandidates.

attributes Objects have internal states or attributes.

methods Objects have functions or methods acting on them.

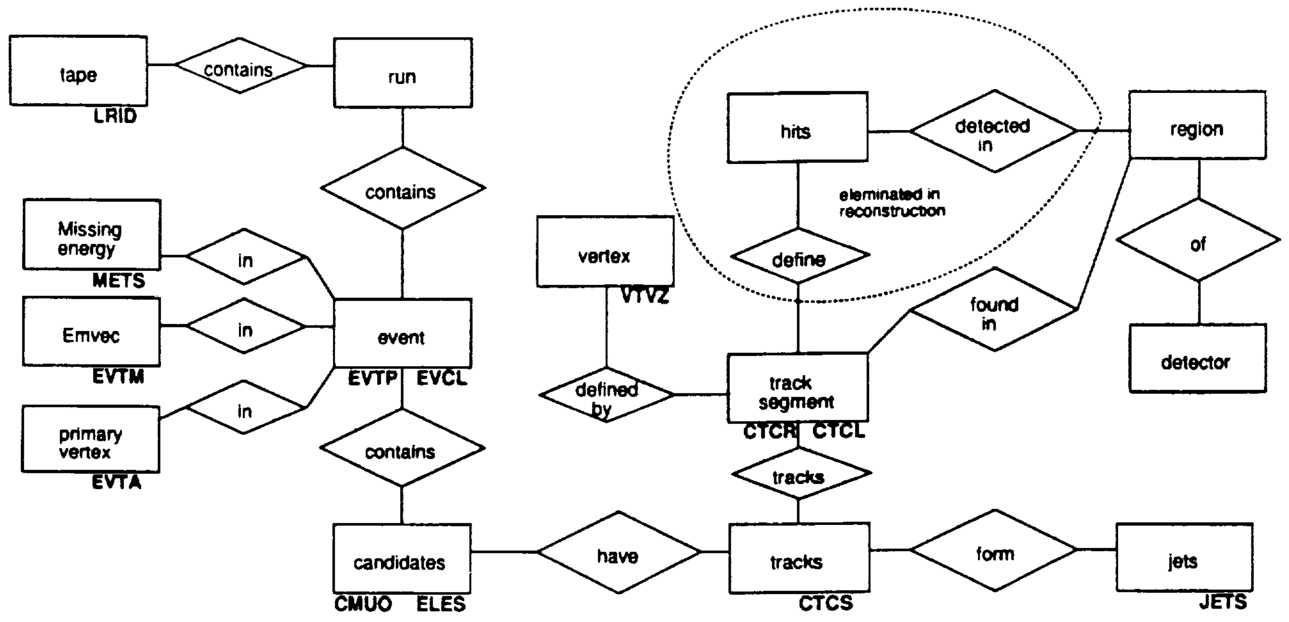

Figure 8: Entity-Relationship Diagram

inheritance Classes can inherit attributes and methods from other classes. The prototype we built above Object Store supports multiple inheritance. The current Mark 1 prototype of the database supports single inheritance.

collections Objects may be grouped together into collections. Typically, this is done by selecting objects from other collections on the basis of their attributes or on the basis of the results of methods acting on them. For example, in Figure 5, NeutralEvents is a collection consisting of objects of type TEvent which contain two leptons of equal, but opposite charge.

derived data A class C is said to be *derived* from one or more other classes D_i, if the values of instances of class D are functionally dependent on instances of the classes C_i. When the values of instances of class D_i are changed, the changes are propagated (either automatically, or after certain triggers) so that instances of class C are recomputed.

5 Discussion

In this section, we make a number of remarks concerning our experience to date in analyzing HEP data using the HEPDBC System.

Remark 5.1 An important advantage of a data model supporting complex objects is the ability to access just the parts of the events necessary to process a given query. This is one of the main reasons for our speed up over the current production version of the physics analysis system.

Remark 5.2 One of the main advantages of system supporting methods is the ability to incorporate Fortran and C functions efficiently into the query and access methods.

Remark 5.3 For the types of queries studied, a traditional transaction model seems inappropriate. By and large, the data is historical, infrequently accessed and almost never written. Most of the computations require only transient structures. The end of a long sequence of computations will be a table or collection that can typically be stored in a "local" database used by a working group. A transaction model may be appropriate for the smaller, local databases.

Remark 5.4 The analysis of HEP data requires that a hierarchical storage system [13] be incorporated seamlessly into the database. Much of the HEP data will be on near-on-line storage systems. It is necessary to access such data in exactly the same way as data in secondary storage.

Remark 5.5 Better tools must be developed to support databases distributed over hundreds of disks. Currently, the CDF experiment we are analyzing contains approximately 1.5TB of data. Our Mark 0 database can analyze in its current form several GB's of data. One problem, for example, arises when the systems table describing the database do not fit into one disk partition.

Remark 5.6 An important aspect of scientific databases has been the proper use of metadata [15]. In our case, this doesn't seem to be a difficulty for two reasons:

- Modeling the data using the objects-subobjects and objects-attributes graphs seems to provide sufficient structure for easy navigation and access to the data.
- The analysis of HEP data proceeds naturally by a sequence of threshold queries gradually reducing the data into smaller and smaller data sets. A catalog of these datasets, together with a list attached to each event of the datasets to which it belongs to seem to be all that is required for the efficient analysis of HEP data.

6 Conclusion and Future Directions

In summary, the Mark 0 HEPDBC System showed that it was possible in principle to perform the off-line analysis of HEP data using database computing. With this experience, a preliminary version of the Mark 1 version of the system has been designed and parts of it are in the early stages of testing. The Mark 1 version of the system is designed to analyze HEP data spread over many disks and stored in a hierarchical storage system and to provide the necessary tools for a collaboration to work with the many datasets used by different working groups during the course of an experiment. To meet these goals, the Mark 1 system is organized into:

EventStore This is a large store for the historical data of the experiment. The data is stored in a hierarchical storage system. The EventStore is infrequently updated.

EventDatabases Each working group makes uses of an EventDatabase, which is obtained by a database administrator making suitable queries on the historical EventStore. Working physicists then analyze the data using appropriate tools accessing the EventDatabase.

Although the same data model and query language is used for both parts of the Mark 1 system, there are important differences:

- The EventStore is closer to a file sytem than a database: since the data is historical, concurrency controls, facilities for back up, recovery, schema consistency, etc. are restricted to the EventDatabase.
- Due to the expense of queries to the EventStore, only database administrators are allowed to query it. Tools to cost the expense of queries will also be needed.

References

[1] Z. Qian, et. al., "Use of the Adamo data management sytem with Aleph," *Computer Physics Commnications,* Volume 45, 1987, pp. 283–298.

[2] A. Baden and R. Grossman, *A Model for Computing at the SSC,* Superconducting Super Collider Laboratory Technical Report No. 288, 1990.

[3] A. Baden and R. Grossman, "Database computing and high energy physics," *Proceedings of Computing in High-Energy Physics 91,* Universal Academy Press, Inc., Tokyo, to appear.

[4] A. Baden, C. Day, R. Grossman, D. Lifka, E. Lusk, E. May, and L. Price, "A data model for computations in high energy physics (preliminary report)" *Laboratory for Advanced Computing Technical Report* Number LAC91-R8, University of Illinois at Chicago, December, 1991.

[5] M. Carey and L. Haas, "Extensible database management systems," *ACM SIGMOD Record,* Volume 19, 1991, pp. 54-60.

[6] R. Hammond and J. L. McCarthy, editors, *Proceedings of the Second International Workshop on Statistical Database Management,* Springer-Verlag, New York, 1983.

[7] P. P. Chen, "The Entity-relationship model—Toward a unified view of data," *ACM Transactions on Database Systems,* Volume 1, 1976, pp.9–36.

[8] J. C. French, A. K. Jones, and J. L. Pfaltz, "Summary of the final report of the NSF workshop on scientific database management," *ACM SIGMOD Record,* Volume 19, 1991, pp. 32–40.

[9] R. Haskin and R. Lorie, "On extending the functions of a relational database systems," in *Proceedings of the ACM SIGMOD International Conference on the Management of Data,* June 1982, pp. 207–212.

[10] W. Kim, "Object-oriented databases: definition and research directions," *IEEE Transactions on Knowledge and Data Engineering,* Vol. 2, 1990, pp. 327–341.

[11] F. Malvestuto and M. Moscarini, "Query evaluability in statistical databases," *IEEE Transactions on Knowledge and Data Engineering,* Volume 2, 1990, pp. 425–430.

[12] V. M. Markowitz and A. Shoshani "Representing object structuers in relational databases: A modular approach," *Lawrence Berkeley Laboratory Technical Report*, No. LBL 28482, January 1991.

[13] "Mass Storage System Reference Model, Version 4" edited by Sam Coleman and Steve Miller, IEEE. to appear.

[14] Maurizio Rafanelli, "Research Topics in Statistical and Scientific Database Management," in *Statistical and Scientific Database Management,* M. Rafanelli, J. C. Klensin, and P. Svensson, editors, Springer-Verlag, Berlin, 1989.

[15] A. Shoshani, "Properties of statistical and scientific databases," *Lawrence Berkeley Laboratory Technical Report*, No. LBL 29900, November, 1990.

[16] M. Stonebraker, "Managing persistent objects in a multi-level store," in *Proceedings of the 1991 ACM SIGMOD, SIGMOD Record*, Volume 20, ACM, New York, 1991, pp. 2–11.

Affiliations and Support

This project is supported in part by the Office of Energy Research, U.S. Department of Energy.

Andrew Baden, University of Maryland

Chris Day, Lawrence Berkeley Laboratory

Robert Grossman, University of Illinois at Chicago. Roberg Grossman's work was supported in part by NASA Grant NAG 2-513, and the Laboratory for Advanced Computing, University of Illinois at Chicago.

Dave Lifka, Argonne National Laboratory.

Ewing Lusk, Argonne National Laboratory. Ewing Lusk's work was supported by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

Edward May, Argonne National Laboratory.

Larry Price, Argonne National Laboratory.